Event-Based Approach to Modeling Dynamic Architecture: Appliocation to Mobile Ad Hoc Network

J. Christian Attiogbe

Event-Based Approach to Modelling Dynamic Architecture: Application to Mobile Ad-Hoc Network

Christian Attiogbé

LINA - UMR CNRS 6241 - University of Nantes
F-44322 Nantes Cedex, France
Christian.Attiogbe@univ-nantes.fr

Abstract. We describe an event-based approach to specifiy systems with dynamically evolving architecture; the study is illustrated with the structuring and routing in Mobile Ad-hoc Network. The resulting specification is augmented with desired properties and then analysed using theorem proving and model checking tools.

Keywords: Specification, Verification, Dynamic Architecture, Event B.

1 Introduction

Distributed systems modelling, design, analysis and implementation are difficult engineering tasks. They still pose challenging specification and analysis difficulties. To master them one needs specific languages, methods and tools.

The general motivation of our work is the need for practical methods, techniques and tools to help the developers in specifying and analysing asynchronous systems with dynamically evolving architecture. They are systems composed of several processes (multi-process) but their number and their structure may be varying in the time. In this article we focus on the systematic specification and analysis of these multi-process systems with evolving structure. We use Mobile Ad-hoc Networks (MANET) as application domain. The expression *dynamic architecture* refers to the evolving structure of such systems.

The contribution of this work is twofold: *i)* an event-based method to guide the specification and analysis of multi-process systems that have dynamic architecture; *ii)* a proof of concept on mobile ad-hoc network modelling and analysis.

The remainder of the article is organised as follows: in the section 2 we describe the main features of dynamic architectures and we present our specification method. Section 3 provides an overview of the used tools (Event B and Pro B). Section 4 presents the modelling and analysis of MANET. Finally Section 5 concludes the article.

2 Modelling Dynamic Architecture

In many specification contexts, one has to deal with dynamic configuration of the system architecture : an example is the growing number of client processes that participate in a resource allocation system and that interact with the resource server.

T. Margaria and B. Steffen (Eds.): ISoLA 2008, CCIS 17, pp. 769–781, 2008.

2.1 Features of Multi-process Systems

Two main features characterise systems with dynamic architecture: *structuring* and *interaction*.

The structure of a classical centralised software system is based on the composition of several sub-systems or processes. They are often parallely composed to enable synchronisation and communication. Unlikely, decentralised systems with dynamically evolving architecture have unfixed but varying structure. They cannot be structured with parallel operators that compose a fixed number of processes; they have an ad-hoc structure related to the number of involved processes.

Interaction is supported by communication and synchronisation between a group of processes currently involved in the cooperation to achieve given goals (the ones defined at the global system level). A group communication is then needed for systems with dynamic architecture. But the structure of the group, hence the architecture of the system, is varying; processes may join or leave the group at any time. The interaction among the processes that compose the system is based on message passing. A process of a group may send/receive messages to/from other processes of the group. Regarding approaches such as finite state automata, multi-process systems are often dealt with by considering the composition or reasoning on an arbitrary high number of processes. However, it is a biased solution to the problem of dynamic architecture.

2.2 Related Specification Approaches

State Transitions or FSM Approach. Capturing a process behaviour is intuitive but state transition systems lack high level structures for complex processes. Handling an undefined, variable number of processes is not tractable; dealing with several instances of the same processes is not possible; synchronisation of processes should be made explicit.

Process algebras. (such as CCS [16], CSP[18], LOTOS[15]) generalise state transition approaches and are widely used to model interacting processes; herein the behaviours of elementary processes are described and then the parallel composition operators are used to combine the processes. Therefore the architecture of a system is also a static composition of a finite number of processes. The $\pi-$calculus [17] permits the description of evolving structures of processes but new processes are generated from existing ones with the name passing mechanisms; the $\pi-$calculus is also not yet well supported by tools.

Handling dynamic behaviour of processes and their architecture is not well treated with the above classical approaches. Event-based approaches provide solutions, they do not consider a specific configuration of communicating processes. Events may be guarded and their occurrence may impact on any process of the current system architecture.

B System Approach. The Event B approach is an event-based one where communicating asynchronous systems are modelled with the interleaved composition of their behaviours viewed as event occurrences. A difficult concern is that of the

completeness with respect to event ordering (liveness concerns): did the specification cover all the possible evolution (event sequences) expressed in the requirement? Indeed one can have a consistent system (with respect to the stated invariant) which does not meet the desired behavioural requirements. This is particularly challenging for dynamically evolving systems.

Therefore rigorous guidelines are needed to help in discovering and expressing the desired behaviours of a system with dynamic architecture; liveness properties help to cover the related completeness aspect. The approach [5] that is used here combines a process-oriented view (at low level) and an event-based one (at global level); it copes with the specification of the dynamically interacting processes and deals with the limitations described above. As an experimental framework we use the Event-B method.

2.3 The Specification Method: Overview

The used specification method is summarised as follows.

- Structuring aspects: each identified *type of process* P_i that may participate in the global system model is specified by considering its space state S_i and the events E_i with their description Evt_i that leads its behaviour and the events to join and leave the system. Note that some events are common to several processes; they handle interaction ans sharing aspects.

  $$P_i \mathrel{\widehat{=}} \langle S_i, E_i, Evt_i \rangle$$

  At this low level, a process-oriented view is consider to discover the needed events for a process behaviour.
- Interaction aspects: as far as communication is concerned we use guarded events, message passing and ordering event occurrences; the processes synchronise and communicate through the enabling/disabling of the guards of their events. An event is used to model a process which is waiting for a data; it may be blocked until the availability of the data (enabling the event guard), which is the effect produced by another process event. Consider for example the case of processes exchanging messages, one process waits for the message and the other process sends the message. An abstract channel modelled as a set, is used to wait for a message or to deposit it. Hence the interaction between the processes are handled using common abstract channels. By the way, the communication is achieved in a completely decoupled way to favour dynamic structuring.
- All the described processes are combined by a fusion operation that merges state spaces and the events of the processes into a single global system S.

  $$S \mathrel{\widehat{=}} \biguplus_i \langle S_i, E_i, Evt_i \rangle$$

In the following the method is illustrated with the MANET system using B abstract system.

3 Overview of the Used Materials

In this study we use the Event B method as the practical framework of our specification method of the MANETs. Prior to the formal specification we provide an overview of the Event B method[2,4] and the related Pro B tool[13].

3.1 Overview of Event B

Within the Event B framework, asynchronous systems may be developed and structured using *abstract systems* [2,4]. *Abstract systems* are the basic structures of the so-called *event-driven* B, and they replace the *abstract machines* which are the basic structures of the earlier *operation-driven* approach of the B method[1]. An *abstract system* [2,4] describes a mathematical model of a system behaviour[1]. It is made mainly of a state description (constants, properties, variables and invariant) and several *event* descriptions. Abstract systems are comparable to Action Systems [7]; they describe a nondeterministic evolution of a system through guarded actions. Dynamic constraints can be expressed within abstract systems to specify various liveness properties [4,10]. The state of an abstract system is described by variables and constants linked by an invariant. Variables and constants represent the data space of the system being formalised. Abstract systems may be refined like abstract machines [10,3].

Data of an Abstract System. At a higher level an abstract system models and contains the data of an entire system, be it distributed or not. Abstract systems have been used to formalise the behaviour of various (including distributed) systems [2,9,10,3]. Considering a global vision, the data that are formalised within the abstract system may correspond to all the elements of the distributed system.

Events of an Abstract System. Within B, an event is considered as the observation of a system transition. Events are spontaneous and show the way a system evolves. An event e is modelled as a *guarded substitution*: $e \mathrel{\widehat{=}} eG \Longrightarrow eB$ where eG is the event *guard* and eB the event *body* or *action*.

An event may occur or may be observed only when its guard holds. The action of an event describes, with generalised substitutions, how the system state evolves when this event occurs. Several events may have their guards held simultaneously; in this case, only one of them occurs. The system makes internally a nondeterministic choice. If no guard is true the abstract system is blocking (deadlock).

An event has one of the general forms (Fig. 1) where gcv denotes the global constants and variables of the abstract system containing the event; bv denotes the bound variables (variables bound to ANY). $P_{(bv,gcv)}$ denotes a predicate P expressed with the variables bv and gcv; in the same way $GS_{(bv,gcv)}$ is a generalised substitution S which models the event action using the variables bv

[1] A system behaviour is the set of its possible transitions from state to state beginning from an initial state.

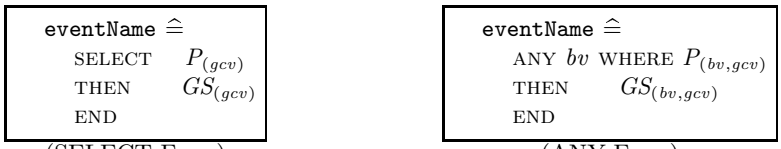

Fig. 1. General forms of events

and gcv. The SELECT form is a particular case of the ANY form. The guard of an event with the SELECT form is $P_{(gcv)}$. The guard of an event with the ANY form is $\exists(bv).P_{(bv,gcv)}$.

Semantics and Consistency. The semantics of a B model described as an abstract system relies on its invariant and is guaranteed by proof obligations (POs). The *consistency* of the model is established by such proof obligations:

i) the initialisation U should establish the invariant I: $[U]I$;
ii) each event of the given abstract system should preserve the invariant of the model.

The proof obligation of an event with the ANY form (Fig. 1) is:

$$I_{(gcv)} \wedge P_{(bv,gcv)} \wedge \mathsf{term}(GS_{(bv,gcv)}) \Rightarrow [GS_{(bv,gcv)}]I_{(gcv)}$$

where $I_{(gcv)}$ stands for the invariant of the abstract system.

The predicate $\mathsf{term}(GS_{(bv,gcv)})$ expresses that the event should terminate. The deadlock-freeness should be established for an abstract system: the disjunction of the event guards should be true. The event-based semantics of an abstract system A is the event traces of A ($traces(A)$); the set of finite event sequences generated by the evolution of A. The B method is supported by the theorem provers Atelier-B [12] and B-Toolkit [6] which are industrial tools. Public domain tools such as B4free[2] and ProB[3] are available.

3.2 Overview of ProB

The ProB tool [13,14] is an animator and a model checker for B specifications. It provides functionalities to display graphical view of automata. It supports automated consistency checking of B specifications (an abstract machine or a refinement with its state space, its initialisation and its operations). The consistency checking is performed on all the reachable states of the machine. The ProB also provides a constraint-based checking; with this approach ProB does not explore the state space from the initialisation, it checks whether applying one of the operation can result in an invariant violation independently from the initialisation.

The ProB offers many functionalities. The main ones are organised within three categories: *Animation*, *Verification* and *Analysis*. Several functionalities

[2] B4free is one of the tool dedicated to Event B: `www.B4free.fr`
[3] ProB `www.stups.uni-duesseldorf.de/ProB/`, is a free model checker for B.

are provided for each category but here, we just list a few of them which are used in this article.

In the *Verification* category, the following functionalities are available:
Temporal Model Checking: starting from a set of initialisation states (initial nodes), it systematically explores the state space of the current B specification.
LTL Model Checking: this functionality enables one to check the specification against a given LTL property.

In the *Analysis* category we consider the following functionality:
Compute Coverage: the state space (the nodes) and the transitions of the current specification are checked, some statistics are given on deadlocked states, live states[4], covered and uncovered operations.

The ProB tool is used in our study to help in discharging consistency proof obligations (invariant violation) and to check liveness properties.

4 Modelling the MANET System

The study of MANET (Mobile Ad-hoc Network)[11] is an active and challenging field as this type of network is rapidly growing and supporting small and medium size applications such as mobile services sharing, wireless peer-to-peer systems, etc. We chose the field of MANET for this work because it is a challenging field in the frontier of computer networks and software engineering. Especially, communication protocols, which are specific software systems, should be correct to ensure the (quality of) services deployed on networks. From the software system point of view, the MANET system is a typical asynchronous system with dynamically evolving architecture, it is decentralised. Moreover, its properties (dynamicity, mobility, correctness, etc) need a combined use of several verification techniques (namely a multifacet analysis).

4.1 Overview of Mobile Ad-Hoc Network

A mobile ad-hoc network [11] is a network formed by wireless mobile nodes (called ad-hoc nodes) which are the users equipments or devices. A MANET has no dedicated network infrastructure, but each node serves as a part of the network and acts a *router* to forward messages or packets since there is no router dedicated to that task.

A mobile ad-hoc network is formed only when a group of users put together their resources to enable and perform communications; hence a mobile ad-hoc network is dynamically created and may also desappear quickly.

In a MANET, the nodes communicate either by exchanging directly or via intermediate nodes. Technically they use ISM band[5] and more generally Wireless LAN technologies. Each node is equipped with one or more radio interfaces with specific transmission features. The *transmission range* of a node is the transmission area accessible from this node. All the nodes in this range are

[4] The already computed states.

[5] They are radio system frequency initially dedicated to industrial, scientific and medical usage.

accessible directly (one hop); they are called the neighbours. To address a known node which is not in its transmission range, the sender node sends its packet to one of the neighbour nodes which is closer to the destination node (according to the transmission ranges). Each node may communicates directly or indirectly using relay nodes (multi-hop), with other nodes that are outside the sender range.

Dynamic Aspect. One of the main features of a MANET is its dynamic aspect: the structure or topology of the network is frequently changing. A node may join or leave the net at any time, changing the net topology. The structure or topology of the net is then highly dynamic.

Mobility Aspect. The ad-hoc nodes may move at any time and very frequently due to their mobile nature; consequently this impacts not only on the net topology but also on its quality; there may be route changes, information loss, partitions of the network into different networks, etc. As far as routing is concerned, in classical infrastructure-based network, there are one or several nodes called routers that are in charge of routing packets between nodes. For this purpose the routers and the nodes are equipped with a routing table where there is the information about how to join a given destination node or a network identified with an Internet Address (IP address).

In the scope of MANET, efficient routing protocols development is a challenging concern. A message or packet sent to a node reaches it unless the net is partitionned. The destination node of a packet is either in the range of the sender node or it is in the range of an intermediate node that is closer to the destination node or that is itself the destination. Concerning the time, it is assumed to be discrete and divided into frames. A node has a set of neighbour nodes during a frame. During a frame a node may be iddle, it also may send messages, receive messages, forward the received messages. Before sending a message to a destination, a source node *sn* which does not have the destination node address, sends a route request to get this destination address. The request travels through the net possibly with multi-hop and reaches the destination which sends back it address. When the address is received by *sn* the latter can send its message to the right destination address.

4.2 Formal Specification of MANET

In our study, a MANET is viewed as an evolving global system. Formally, it is a set of nodes with a connection relationship: a configuration. The evolution of the MANET is viewed as the combined evolution of the nodes, hence a sequence of configurations; going from a configuration to another is observed as an event and it depends on the actions performed by the net nodes.

Specifying Node Processes. A node is modelled as a process using Event B. Each node has some features: an identifier, a location, an IP address, a connection relation that indicates its neighbours, etc. Accordingly we have the S_i part of the node. A set of events (E_i) with the associated behaviours (Evt_i) defines the process behaviour which leads the evolution of the system. Any node may initiate a message for a given destination, send a message, receive a message, forward a

message, leave a net (a transmission range). The behaviour described by these events is observed only when a net exists; that means the net structuring events are related to those needed for the routing. Also we deal with the creation of a network by nodes which have a given range, other nodes may join or leave this range. Therefore, in the B model, we link the range of a node with a given abstract network.

Event B Specification of MANETs. The MANET is formed by the nodes (already defined with S_i, E_i, Evt_i). The formal specification of a MANET is a set of possible sequence of configurations of the considered nodes. Concerning the structuring aspect, we describe the configuration by *state variables* (hence a state space) resulting from the fusion of the node state variables; the sequence of configurations is modelled through the enabling of *events* which possibly modify the state space. Concerning the evolution of the entire MANET system, we consider the events of the nodes and also the common events related to the entire system network (*ie* the management of ranges). All the network is dynamic, the nodes leave and join it at any time, new ranges appear, others disappear, etc.

Moreover, from the methodological point of view, we have considered two aspects in the Event B specification of MANET: the structuring of the networks (the configuration related to the net topology) and the routing in the networks.

As far as routing is concerned we consider one of the widely studied routing protocol of MANET: *Ad-hoc On demand Distance Vector* (AODV) [11].

Therefore a part of our B specification is related to the structuring and another part is about the routing protocol.

Specifying the MANET Structure. The structuring of a MANET is achieved using a set of state variables and an invariant that describes the nodes and their current configurations:

INVARIANT
$nodes \subseteq NODE \wedge ranges \subseteq RANGE \wedge messages \subseteq MSG$
$\wedge\ rangNodes \in ranges \leftrightarrow nodes \wedge reqMsg \in nodes \leftrightarrow messages$
$\wedge\ inReqMsg, inRspMsg \in nodes \leftrightarrow messages$
$\wedge\ waitReqMsg \in nodes \leftrightarrow messages$
$\wedge \cdots$

The evolution of the system depends on the set of events that define the nodes and the specific system events: the observation of a net creation (*newRange*); an existing net may disappear if there is no more connected nodes (*rmvRange*). The other events considered for the network structuring are summarised in the table Tab. 1;

The combination of the two categories of events forms an abstract MANET specification which is the reference model for the specification. It describes a system composed of node processes and abstract MANET networks. The evolving of the system architecture is based on the fact that the event guards depends on the variables $nodes, messages, \cdots$ which in turn depend on current event. This is illustrated by the non-deterministic form of the event specifications:

$$\texttt{event} \mathrel{\widehat{=}} \text{ANY } sn \text{ WHERE } sn \in nodes \text{ THEN ... END}$$

Table 1. Network structuring events

Event	Description
newRange	A new network range appears
joinRange	A node joins a range
leaveRange	A node leaves a net range
newNode	A new node appears
newMsg	A node initiate a message

Specifying the AODV Routing Protocol. Within the Ad-hoc On demand Distant Vector (AODV) protocol, each node acts as a router, contributes to construct routes and forward messages to other nodes. There are two phases of the protocol: route discovery and route maintenance. Route discovery is achieved by exchanging Route Request (RREQ) and Route Response (RREP) messages. The algorithm of the nodes is as follows: when a node desires to set up a route to a destination node, it broadcasts a RREQ message to its neighbours (the nodes in its range). The RREQ/RREP messages have the following main parameters: the source node Id, the destination node Id, the number of hop.

When a node *nd* receives a RREQ message, *i)* either *nd* is itself a destination and *nd* responds with a RREP or *nd* is an active route to the searched destination node then *nd* responds with a route information using the RREP message; *ii)* otherwise *nd* broadcasts the RREQ further with the hop count of RREQ increased by 1. When a node *nd* receives a duplicate RREQ, it drops the message. The routing of message is symmetric when a node receives a RREP message. The Event B specification comprises the events related to the routing protocol described above. These events are listed in the table Tab. 2.

Table 2. Routing events

Event	Description
sndRREQ	Route Request sending
fwdRREQ	Route Request forwarding
rcvRREQ	Route Request receiving
sndRREP	Route Response sending
fwdRREP	Route Response forwarding
rcvRREP	Route Response receiving

The B specification of a MANET is then an abstract system equipped with these events (see Fig. 2).

We give in the following (see Fig. 3) the specification of the `sndRREQ` event to illustrate the specification principle. Here, any node (*sn*) may send a message (*msg*) that it has already prepared ($msg \in reqMsg[\{sn\}]$) to all the nodes in its range (*otherNodesInRange*). Exchanged messages are modelled using abstract channels (*inRepMsg*,*repMsg*).

The other events are specified in quite the same way. Therefore the complete specification enables us to model the dynamic evolution of the MANET (as

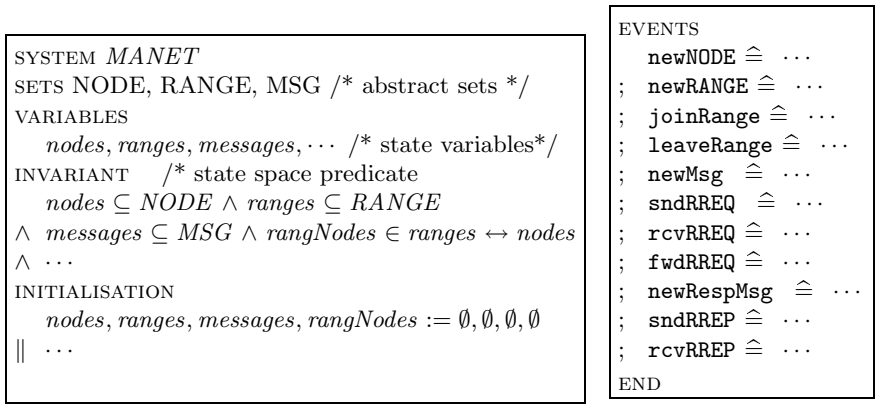

Fig. 2. Structure of the abstract system

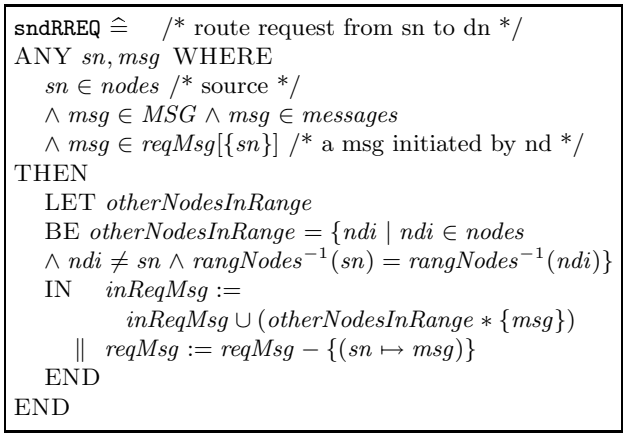

Fig. 3. Specification of the `sndRREQ` event

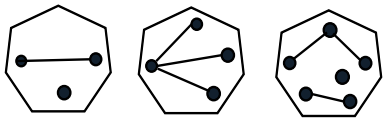

Fig. 4. Evolution and various dynamic interactions

illustrated in Fig. 4) and the routing protocol via dynamically interacting variable number of node processes.

4.3 Analysis of the Specified MANET System

A multifacet analysis with a reference abstract model is performed on the MANET system. For this purpose two different tools are used but they cover different facets

of the analysis: B4free and ProB [13]. Both tools use one common input specification: the B reference model previously specified; this ensures consistency of verification and feedbacks.

Consistency and Refinement of System. The previously described abstract system is proved consistent (see Sect.3.1) using the B4free tool. Then it is refined; more details are added to the state space and the event specifications; for instance we consider the management of the IP addresses of the nodes and exchanged messages. Unlike in the abstract system where a packet destination is nondeterministically selected, in the refinement the nodes and the messages have IP addresses, therefore, the receiver node is checked against the destination IP address. The resulting refined system is also proved correct with respect to consistency using the B4free tool. However to accomplish the proofs, we combine the use of B4free and ProB. That is, when a proof obligation is not discharged by B4free, we model-check the specification and discover possible errors by displaying and analysing the displayed error state. Accordingly the feedback is propagated in the reference model and we iterate. This multifacet analysis approach helps here to make precise the correct ordering of the events: the simulation functionalities and the listing of uncovered operations help to correct the B abstract system. This aspect is very important because, an abstract system proved correct, may have an incomplete or even a wrong behaviour if for example we have an event which is never enabled. Using the multifacet approach, helps us to get a complete analysis. The ab. 3 shows a ProB experiment result where one deadlock is detected after the exploration of 31257 nodes and 1168 transitions ; all operations (the B events) are covered, with the indicated occurrences.

Table 3. Analysis results

NODES	
invariant_violated	: 0
deadlocked	: 1
live	: 2521
explored_transitions	: 1168
open	: 28735
total	: 31257
TOTAL_OPERATIONS	
44110	

COVERED_OPERATIONS	
initialise_machine	: 1
newRANGE	: 225
rcvRREP	: 14
sndRREP	: 29
newRespMsg	: 300
sndRREQ	: 1829
rcvRREQ	: 1697
newNODE	: 10487
joinRange	: 7411
leaveRange	: 9721
newMsg	: 11042
fwdRREQ	: 1354
UNCOVERED_OPERATIONS	

The state corresponding to the deadlock is carefully analysed. We discover that it corresponds to a situation (net partitioning) where there are nodes with some packets to be transmitted but no node in the current net range. This

corresponds to a real-life situation which is due to the dynamic aspect of the MANET and the mobility of nodes. A feedback is then propagated in the Event B specification. To confirm that, the model is corrected by strengthening the guard of message initiation by the hypothesis of non-emptiness of the net range. Thus the analysis of the model runs without errors[6]. In the real-life situation, this corresponds to the fact that after a while the net may be reconstituted with other nodes.

Liveness Properties Analysis. Many properties of the MANET routing protocol are well-expressed using LTL formula which is not supported by the B4free tool. We express these liveness properties with the ProB LTL formalism. Then we extend the Event B abstract system with these LTL properties; the resulting specification is model-checked.

The following are illustrations of some checked properties.

$\mathbf{P}_1$. A route request is always followed by a response:
$G(e(sndRREQ) \Rightarrow F(e(sndRREP)))$ false

$\mathbf{P}_2$. A route request may be followed by a response:
$e(sndRREQ) \Rightarrow F(e(sndRREP))$ true

$\mathbf{P}_3$. A route request may be finally received:
$F(e(sndRREQ) \Rightarrow X(e(rcvRREQ)))$ true

We come to the conclusion that our model of the MANET extended with the stated properties, is correct with respect to these properties.

5 Conclusion

We presented the main features of decentralised system with dynamically evolving architecture; we showed that these features are not well handled with classical state-oriented approaches and accordingly we presented a method that deals with them using event-based approach. The composition of processes used to model the system components is completely decoupled to favour the evolving of the system architecture. The method which combines a process-oriented view (at low level) and an event-based one (at global level) was illustrated with the specification and the analysis of a MANET system. The proof is given that the specified system with dynamic architecture may be studied with respect to safety and liveness properties. For this purpose the Event B tools are used. There are several works on dynamic and self-managing component architectures, [8] presents a survey; most of them use a process-algebra oriented approach, focus on the changes on defined architectures and define rules to perform reconfiguration. Compared with these works our event-based approach adds distribution and mobility of processes and no predefined reconfiguration rules are needed, instead we consider the behaviour of process types. Ongoing works are about the scalability of our approach; we consider precisely two aspects, one is the analysis of Mobile Linux codes (drivers) for embedded systems by considering their abstractions, the other one is the strengthening of message passing aspects and the refinement of our specifications into executable codes for physical devices.

[6] The experiment result tables, not displayed here, show 0 deadlocked states for hundreds of explored states and transitions.

References

1. Abrial, J.-R.: The B Book. Cambridge University Press, Cambridge (1996)
2. Abrial, J.-R.: Extending B without Changing it (for developping distributed systems). In: Habrias, H. (ed.) Proc. of the 1st Conf. on the B method, France, pp. 169–190 (1996)
3. Abrial, J.-R., Cansell, D., Mery, D.: Formal Derivation of Spanning Trees Algorithms. In: Bert, D., et al. (eds.) ZB 2003. LNCS, vol. 2651, pp. 457–476. Springer, Heidelberg (2003)
4. Abrial, J.-R., Mussat, L.: Introducing Dynamic Constraints in B. In: Bert, D. (ed.) B 1998. LNCS, vol. 1393, pp. 83–128. Springer, Heidelberg (1998)
5. Attiogbé, C.: Multi-process Systems Analysis using Event B: Application to Group Communication Systems. In: Liu, Z., He, J. (eds.) ICFEM 2006. LNCS, vol. 4260, pp. 660–677. Springer, Heidelberg (2006)
6. B-Core. B-Toolkit, UK (consulted, 2007), `www.b-core.com`
7. Back, R., Kurki-Suonio, R.: Decentralisation of Process Nets with Centralised Control. In: Proc. of the 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, pp. 131–142 (1983)
8. Bradbury, J.S., Cordy, J.R., Dingel, J., Wermelinger, M.: A survey of self-management in dynamic software architecture specifications. In: WOSS 2004: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems, pp. 28–33. ACM, New York (2004)
9. Butler, M., Walden, M.: Distributed System Development in B. In: Habrias, H. (ed.) Proc. of the 1st Conference on the B method, France, pp. 155–168 (1996)
10. Cansell, D., Gopalakrishnan, G., Jones, M., Mery, D.: Incremental Proof of the Producer/Consumer Property for the PCI Protocol. In: Bert, D., P. Bowen, J., C. Henson, M., Robinson, K. (eds.) B 2002 and ZB 2002. LNCS, vol. 2272, pp. 22–41. Springer, Heidelberg (2002)
11. Chlamtac, I., Conti, M., Liu, J.: Mobile Ad hoc Networking: Imperatives and Challenges. Ad Hoc Networks 1(1), 13–64 (2003)
12. ClearSy. Atelier B V3.6. Steria, Aix-en-Provence, France, (consulted, 2007), `www.clearsy.com`
13. Leuschel, M., Butler, M.: ProB: A Model Checker for B. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 855–874. Springer, Heidelberg (2003)
14. Leuschel, M., Turner, E.: Visualizing Larger State Spaces in ProB. In: Treharne, H., King, S., C. Henson, M., Schneider, S. (eds.) ZB 2005. LNCS, vol. 3455, pp. 6–23. Springer, Heidelberg (2005)
15. Lotos, I.: A Formal Description Technique Based on The Temporal Ordering of Observational Behaviour. In: IOS - OSI, Geneva (1988); International Standard 8807
16. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
17. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes. Journal of Information and Computation 100 (1992)
18. Roscoe, A.: The Theory and Practice of concurrency. Prentice-Hall, Englewood Cliffs (1998)